REMARKS

This application has been carefully considered in connection with the Final Office Action dated August 20, 2008. Reconsideration and allowance are respectfully requested in view of the following.

Summary of Rejections

Claims 1-25 were pending at the time of the Office Action.

Claims 1-25 were rejected under 35 USC § 102.

Summary of Response

Claims 1, 5, 7, 15-16, 18, 19 and 24 were previously presented.

Claims 2-4, 6, 8-14, 17, 20-23 and 25 remain as originally submitted.

Remarks and Arguments are provided below.

Summary of Claims Pending

Claims 1-25 are currently pending following this response.

Examiner Initiated Interview

Applicants thank Examiner John Anderson for his time in the telephone interview on September 23, 2008. In the interview the Examiner indicated that after further consideration of the applied art, the alleged 102 rejection was improper. The Examiner

also indicated that the finality of the previous office action will be withdrawn, and a subsequent new rejection will be issued.

Response to Rejections

Bowman-Amuah does not disclose streaming data conversion that extracts data from one system, converts the data into a format compatible with a second system, and loads the converted data into the second system in real-time. Rather than running sequentially as in a conventional batch process, the streaming data conversion processes disclosed in the pending application perform the extracting, converting and loading of data generally in parallel, which improves the overall streaming performance of the system involved.

The pending application discloses systems and methods for streaming conversion of data. For example, the disclosed streaming conversion methods can be used to convert customer billing data stored in a legacy system format to a new billing system format and load the converted data in the new billing system format in real-time. As opposed to conventional techniques that use batch processing, the disclosed streaming conversion processes are performed generally in parallel, which minimizes system outages and customer down-time.

Bowman-Amuah is directed to a system, method and article of manufacture for translating an object attribute to and from a database value. In general, Bowman-Amuah discloses a development architecture framework for constructing and

maintaining application software. Bowman-Amuah generally discloses a development architecture framework useful for software application batch processing, and certain implementations, such as object-oriented-type programming for converting object attributes to database values. However, Bowman-Amuah does not disclose, teach or suggest streaming data conversion that extracts data from one system, converts the data into a format compatible with a second system, loads the converted data into the second system, and performs the extracting, converting and loading generally in parallel, as claimed.

These distinctions, as well as others, will be discussed in greater detail in the analysis of the present claims that follows.

Response to Rejections under Section 102

Claim 1:

Claim 1 was rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah, U.S. Patent No. 6,529,909 ("Bowman-Amuah").

I. Bowman-Amuah does not disclose an extractor component that extracts a unit of data from the first system.

Claim 1 recites, "an extractor component that extracts a unit of data from the first system."

Bowman-Amuah's framework is intended to provide an inventory of software components needed to design, build, install and operate the systems involved, and an understanding of how the components should fit together conceptually. (Bowman-Amuah Col. 20, lines 14-20). Bowman-Amuah describes the benefits of its architectural approach to software development as increased productivity and "less reinvention of the wheel." (Bowman-Amuah Col. 20, lines 45-53). As a specific implementation of its architectural approach to software development, Bowman-Amuah discloses a method for translating an object attribute to and from a database value. Bowman-Amuah describes its method as a type of object-oriented programming and defines an "object" as a self-sufficient software package that contains both data and a collection of related structures and procedures, and does not require other additional structures, procedures or data to perform its specific task. (Bowman-Amuah Col. 192, lines 47-59). The method determines a conversion process to be used for converting the object attribute to and from a database value. The conversion process is then encapsulated in an attribute converter. The object attribute is then directed to the attribute converter to be converted to a database value.

The Office Action relied on the following disclosure in Bowman-Amuah (column 20, lines 25-34) to read on the extractor component recited in claim 1:

> Frameworks are used to help practitioners understand what components may be required and how the components fit together. Based on the inventory of components and the description of their relationships, practitioners will select the necessary components for their design. An architect extracts components from one or more

> Frameworks to meet a specific set of user or application requirements. Once an architecture has been implemented it is often referred to as an architecture or an infrastructure. (Underlining added for emphasis.)

As shown above, Bowman-Amuah merely discloses one or more steps that an architect might take to meet a set of user or application requirements. As shown, the word "extracts" is disclosed. However, Bowman-Amuah does not disclose what components would be extracted or how such an extraction might actually be accomplished (e.g., manually, computer-assisted, etc.). Clearly, Bowman-Amuah's disclosure of an architect extracting components from one or more frameworks to meet user or application requirements does not teach or suggest "an extractor component that extracts a unit of data from the first system," as recited in claim 1.

II. Bowman-Amuah also does not disclose a loader component that loads the unit of data converted to the second data format into the second system, and the extractor, the translator, and loader components convert the unit of data during normal operation of the first and second systems, wherein the extractor, translator and loader components extract, convert and load generally in parallel.

Claim 1 recites, "a loader component that loads the unit of data converted to the second data format into the second system, and the extractor, the translator, and loader components convert the unit of data during normal operation of the first and second systems, wherein the extractor, translator and loader components extract, convert and load generally in parallel."

The Office Action relied on the following disclosure in Bowman-Amuah (column 193, line 63 to column 194, line 8) to read on the loader component recited in claim 1:

> Abstraction Factory:
>
> AbstractType produceForKey(key)
>
> Abstract Type:
>
> init(some data stream)
>
> the Abstraction Factory can be fully coded in C++. It is very re-usable as it stands. In addition, it has been extended to perform 'Java Loader-like' dynamic linking if the proper code cannot be found already within the factory.
>
> Factory, the well know[sic] pattern from Gamma, et. al BUW, in which the objects created by the factory can be dealt with generically in terms of independence, scalability, parallel processing, etc. Component Solutions Handbook.

As shown above, Bowman-Amuah describes the use of an Abstraction Factory in an object-oriented programming environment. The above-disclosed section of Bowman-Amuah clearly does not teach or suggest a loader component that loads the unit of data converted to the second data format into the second system, and the extractor, the translator, and loader components convert the unit of data during normal operation of the first and second systems, wherein the extractor, translator and loader components extract, convert and load generally in parallel, as recited in claim 1.

However, in column 192, lines 46-65, Bowman-Amuah describes the details of an Abstraction Factory. In particular, Bowman-Amuah discloses a method for providing an abstraction factory pattern. According to the method disclosed, data is received and

transformed into a plurality of concrete objects. Each of the concrete objects is associated with an abstract interface. A map of the association between the concrete objects and the abstract interface is then created. Thus, when a request for an abstraction pattern is received, the request includes an identifier for one of the concrete objects and an identifier for the abstract interface. The map is then consulted to locate the concrete object that has been identified. An abstract object is then created that corresponds to the located concrete object.

Additionally, in column 193, lines 20-43, Bowman-Amuah further describes its Abstraction Factory. Specifically, Bowman-Amuah discloses:

> ...one transforms the various types of raw data into a corresponding variety of concrete object types, all of which share a common abstract interface. This transformation will be encapsulated within an Abstraction Factory. The primary interface to the Abstraction Factory is:
>
> 'abstractType produceForKey(key)'
>
> where 'abstractType' is the type of the common abstract interface, and key is a piece of information which identifies the appropriate concrete type. ...When this method is invoked, the Abstraction Factory consults its internal mapping and creates an 'empty' object of the proper concrete class. The factory then casts the concrete object into the abstraction and returns it to the method's client. This client (a framework most likely) will then instruct the abstraction to initialize itself from the incoming data stream.
>
> At the end of this process we have an abstract handle to a concrete object which a framework may then manipulate generically.

As shown by all of the above, Bowman-Amuah's disclosure of its "Abstraction Factory" does not teach or suggest the use of a loader component that loads the unit of data converted to the second data format into the second system, and the extractor, the translator, and loader components convert the unit of data during normal operation of the first and second systems, wherein the extractor, translator and loader components extract, convert and load generally in parallel, as recited in claim 1.

Therefore, for at least the reasons established above in sections I and II, Applicants respectfully submit that independent claim 1 is not anticipated by Bowman-Amuah and respectfully request allowance of this claim.

Claims Depending From Claim 1:

Claims 2-6 were rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah.

Dependent claims 2-6 depend directly or indirectly from independent claim 1 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I and II above, Applicants respectfully submit that claims 2-6 are not anticipated by Bowman-Amuah and respectfully request allowance of these claims.

Claim 7:

Claim 7 was rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah.

Claim 7 includes limitations substantially similar to the limitations discussed in sections I and II above. For example, claim 7 recites, "extracting a unit of data from a database associated with the first system; translating the unit of data from a first format accessible by the first system to a second format accessible by the second system; loading the translated unit of data into a database associated with the second system..., wherein the extracting, translating and loading are performed generally in parallel." Therefore, for at least the reasons established above in sections I and II, Applicants respectfully submit that independent claim 7 is not anticipated by Bowman-Amuah and respectfully request allowance of this claim.

Claims Depending From Claim 7:

Claims 8-14 were rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah.

Dependent claims 8-14 depend directly or indirectly from independent claim 7 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I and II above, Applicants respectfully submit that claims 8-14 are not anticipated by Bowman-Amuah and respectfully request allowance of these claims.

Claim 15:

Claim 15 was rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah.

Claim 15 includes limitations substantially similar to the limitations discussed in sections I and II above. For example, claim 15 recites, "an extractor component that extracts a unit of data from the first system; a translator component that converts the unit of data from the first format compatible with the first system to the second format compatible with the second system; a loader component that loads the unit of data converted to the second format into the second system, and the extractor, the translator, and loader components convert the unit of data during normal operation of the first and second systems, wherein the extractor, translator and loader components extract, translate and load generally in parallel." Therefore, for at least the reasons established above in sections I and II, Applicants respectfully submit that independent claim 15 is not anticipated by Bowman-Amuah and respectfully request allowance of this claim.

Claims Depending From Claim 15:

Claims 16-25 were rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah.

Dependent Claims 16-25 depend directly or indirectly from independent claim 15 and incorporate all of the limitations thereof. Accordingly, for at least the reasons

established in sections I and II above, Applicants respectfully submit that claims 16-25 are not anticipated by Bowman-Amuah and respectfully request allowance of these claims.

Conclusion

Applicants respectfully submit that the pending application is in condition for allowance for the reasons stated above. If the Examiner has any questions or comments or otherwise feels it would be helpful in expediting the application, the Examiner is encouraged to telephone the undersigned at (972) 731-2288.

The Commissioner is hereby authorized to charge payment of any further fees associated with any of the foregoing papers submitted herewith, or to credit any overpayment thereof, to Deposit Account No. 21-0765, Sprint.

Respectfully submitted,

Date: October 20, 2008

/Michael W. Piper/
Michael W. Piper
Reg. No. 39,800

ATTORNEY FOR APPLICANTS

CONLEY ROSE, P.C.
5601 Granite Parkway, Suite 750
Plano, Texas 75024
(972) 731-2288
(972) 731-2289 (facsimile)